SAND80-0538/4
Unlimited Release
UC-32

Remote Hard Copy
Volume 4 — MODCOMP-NOVA Interface Programming Manual

Randall Simons

Prepared by Sandia Laboratories, Albuquerque, New Mexico 87185
and Livermore, California 94550 for the United States Department
of Energy under Contract DE-AC04-76DP00789

Printed March 1980

Sandia National Laboratories

SF 2900-Q(3-80)

DISCLAIMER

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation

---

Printed in the United States of America

Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161

Price: Printed Copy $4.50; Microfiche $3

SAND80-0538/4
Unlimited Release
Printed March 1980

REMOTE HARD COPY, VOLUME 4

MODCOMP-NOVA Interface Programming Manual

R. W. Simons
Distributed Processing System Design Division, 2644
Sandia National Laboratories
Albuquerque, New Mexico 87185

ABSTRACT

This manual gives a detailed description of commands accepted by the MODCOMP-NOVA interface used in the remote hard copy plotters. Following that, device drivers and diagnostic routines for the interface are documented.

Contents

Page

Introduction 5

1. Logical Structure 6

1.1 4805 Logical Structure 6
1.2 4040 Logical Structure 6
1.3 4805-to-4040 Register Correspondence 7

2. Instruction Sets 10

2.1 MODCOMP Instructions 10
2.2 NOVA Instructions 12

3. Device Drivers 15

3.1 MODCOMP-Resident Device Driver 15
3.2 NOVA-Resident Device Driver 15
3.3 MODCOMP-NOVA Protocol 22

4. Diagnostics 24

4.1 MODCOMP-Resident Diagnostics 24
4.2 NOVA-Resident Diagnostics 24

References 26

Introduction

The purpose of this manual is to serve as both an introductory and reference manual on the subject of writing programs to use the MODCOMP-NOVA interface. This interface was designed at Sandia Laboratories by Lyndon Pierson, 2648, and Randall Simons, 2644. Its initial application is to provide a link between a MODCOMP II minicomputer being used as a remote job entry terminal (RJET) and a Data General NOVA 3/D minicomputer being used to manipulate and output plot data on an electrostatic plotter.

Physically, the MODCOMP-NOVA interface consists of a 4805 general purpose interface board housed in the MODCOMP, a 4040 general purpose interface board (modified at Sandia) housed in the NOVA, and a cable connecting the two boards. The interface was designed such that all modifications were made on the 4040 board (which has wire-wrap pins and space for extra chips to facilitate such changes). Thus the 4805 is unmodified and the 4040 has been changed to "understand the language" of the 4805.

The MODCOMP-NOVA interface supports the following functions:

MODCOMP-to-NOVA data channel transfers
NOVA-to-MODCOMP data channel transfers
MODCOMP-to-NOVA downline load
MODCOMP-to-NOVA function/command bit transfers
NOVA-to-MODCOMP status bit transfers
MODCOMP-to-NOVA service interrupt
NOVA-to-MODCOMP service interrupt

Comments are invited from readers of this manual who have corrections or other suggestions as to how to improve it. Direct comments to Randy Simons, Distributed Processing System Design Division 2644, 4-3080.

1. Logical Structure

1.1 4805 Logical Structure

The 4805 is addressed using device address 11_{16}. This gives it an I/O priority of 2, data interrupt location 91, service interrupt location D1, and DMP locations TC=63 and TA=73. This puts it in the second I/O group, group B, so I/O instructions should use the suffix "B" and an address of 1. In this manual the mnemonic "MCNV" has been equated to 1 to allow more readable I/O instructions.

The 4805 has a 6-bit command register which is used to send function commands to the NOVA. Due to limitations of the MODCOMP instruction set, only 5 of these bits are actually usable, with the sixth bit determining whether the command register gets loaded on a given instruction. This register can be written to using instructions described in Section 2.1.

The 4805 will cause an interrupt request in the MODCOMP on either of two occasions:

A. A data interrupt (DI) is caused by the completion of a data channel transfer.

B. A service interrupt (SI) is caused in two possible ways:

   1. An external service interrupt comes from the NOVA and indicates that the MODCOMP should check the NOVA's status.

   2. An internal service interrupt occurs when the MODCOMP executes a terminate instruction.

Two status bits are provided to differentiate internal and external service interrupts.

1.2 4040 Logical Structure

The 4040 is addressed using device code 7, which is equivalent to the device mnemonic MCAR. MCAR stands for Multiprocessor Communications Adapter Receiver, which is a Data General product whose conventions we have followed when applicable. However, the 4040 uses priority mask bit 11, instead of 12 (which is standard for MCAR), because the plotter uses bit 12 and the 4040 must be assigned a different priority. In this manual, the mnemonic "MCNV" has been equated to 7 to allow more readable I/O instructions.

A user writing programs on the NOVA will deal directly with 4 registers on the 4040:

Address Register--contains the address in NOVA memory of next data channel transfer word. (16 bits)

Word Count Register--contains the number of words left to be transferred in the next or current data channel transfer. (16 bits)

Function Register--contains commands from the MODCOMP to the NOVA specifying what the NOVA should do next. (8 bits)

Status Register--contains hardware status of the NOVA to be communicated to the MODCOMP. (8 bits, 7 usable)

These registers can be read and written using the commands described in section 2.2.

Besides reading and writing these registers, the only other operations that can be done from the NOVA are to start and stop data channel transfers and to send an interrupt to the MODCOMP.

The 4040 will cause an interrupt request in the NOVA on either of two occasions:

A. A data interrupt is caused by the completion of a data channel transfer.

B. A service interrupt is caused by a change in the value of certain bits in the function register-- receive data block, receive control block, or transmit status.

To differentiate these interrupts, the NOVA will have to check the SI/DI bit in the function register which will equal 0 if the most recent interrupt was a data interrupt or will equal 1 if the most recent interrupt was a service interrupt.

1.3 4805-to-4040 Register Correspondence

The following two diagrams indicate the correspondences that are set up between the specified register in the MODCOMP (Ra), and the function and status registers in the 4040. The MODCOMP instructions Input Status and Output Command cause the actual transfer of bits between these registers.

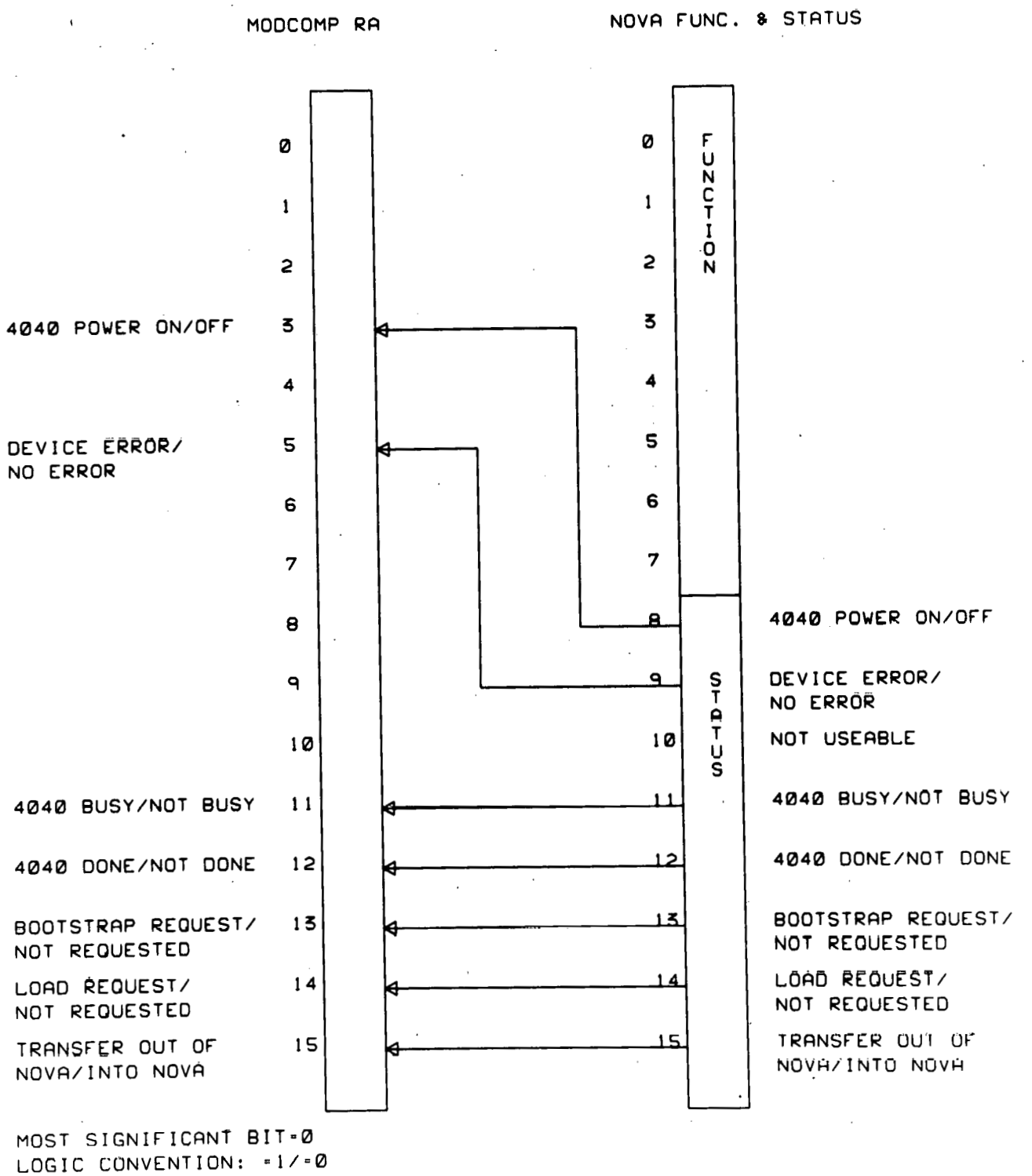

FIGURE 1.
INPUT STATUS

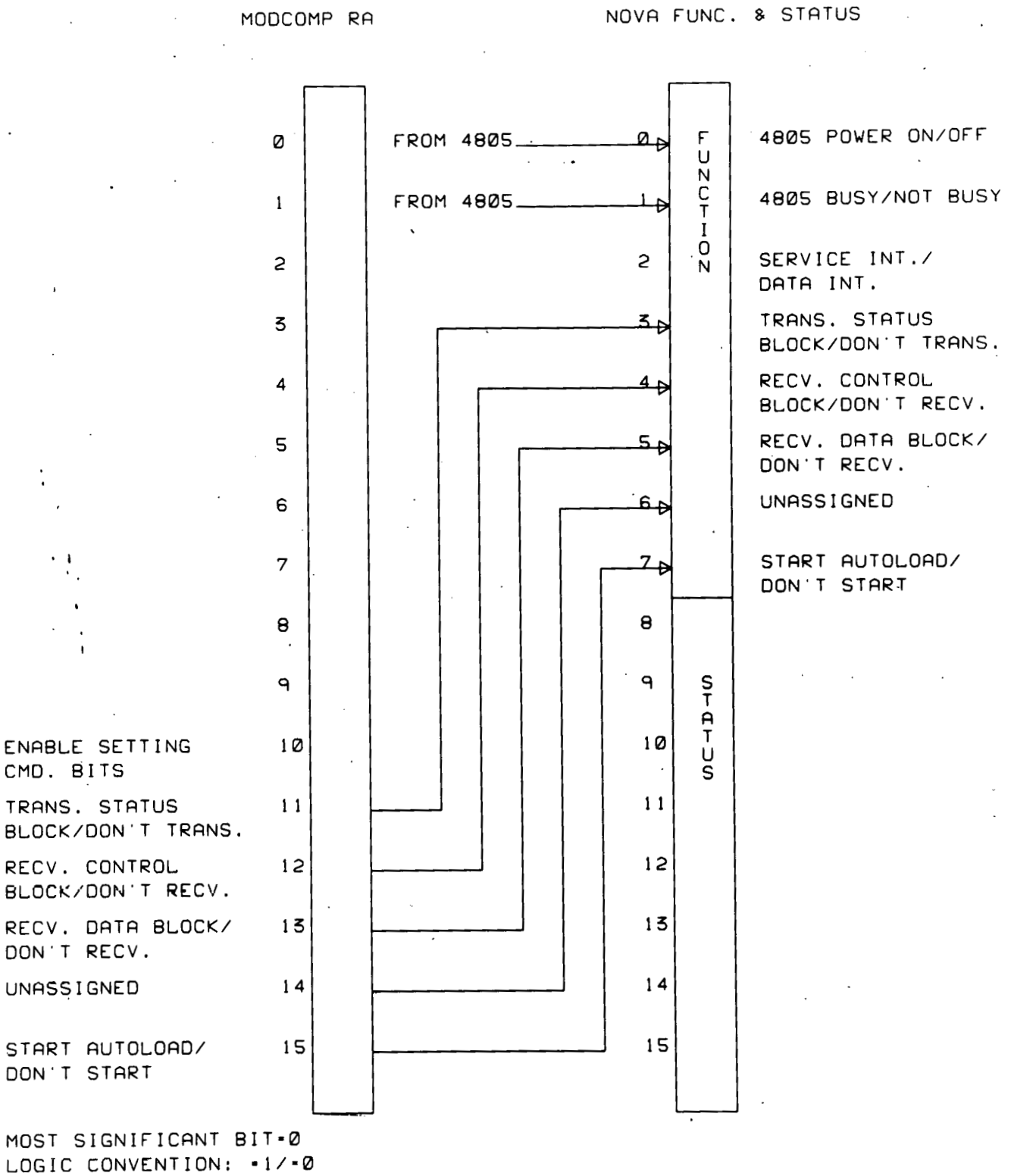

FIGURE 2.
OUTPUT COMMAND

2. Instruction Sets

2.1 MODCOMP Instructions

ISB Ra,MCNV

Input status from the NOVA to register Ra. The 7 status lines coming in to the 4805 and other information are read into Ra. The resultant bits in Ra are assigned the following meanings (most significant bit = 0):

Bit #	Meaning if = 1/=0
0	no error/bit 3, 4 or 5 is set or 4805 not present
1-2	none
3	4040 power on/off
4	MODCOMP DMP memory parity error/no error
5	NOVA device error/no error
6	none
7	4805 busy/not busy
8	MODCOMP data buffer not ready/ready
9	MODCOMP SI caused externally/not external
10	MODCOMP SI caused internally/not internal
11	4040 busy/not busy
12	4040 done/not done
13	NOVA bootstrap request/not requested
14	NOVA load request/not requested
15	transfer out of NOVA/into NOVA

IDB Ra,MCNV

Input data from the NOVA to register Ra. This instruction must be used only when the 4805 is in register mode (set by transfer initiate command), otherwise its effect is undefincd. Only one 16-bit word is input each time this instruction is executed.

OCB Ra,MCNV

Output command from register Ra to the NOVA. The contents of Ra determine what action will be taken by the 4805. There are 4 allowable types of commands, each of which is described below by giving the contents of Ra.

Transfer initiate command:

Bit #	Meaning if = 1/=0
0	=1
1	MODCOMP DMP mode/register mode
2	MODCOMP DI connect/disconnect
3	MODCOMP SI connect/disconnect
4	input to/output from MODCOMP
5-9	none
10	load 4805 command register/don't load command register
11	NOVA transmit status/don't transmit status
12	NOVA receive control block/don't receive
13	NOVA receive data block/don't receive
14	unassigned
15	NOVA start autoload/don't start

If bit 1 equals 1, this will initiate a direct memory processor (DMP) transfer. The DMP locations TA and TC must have already been set to the transfer address and transfer count. These specify where in memory the transfer is to begin, and how many words are in the block to be transferred.

Terminate command:

Bit #	Meaning if = 1/=0
0	=0
1	=1
2	MODCOMP DI connect/disconnect
3	MODCOMP SI connect/disconnect
4	cause a MODCOMP DI (automatic at end of DMP block)
5	terminate 4805/don't terminate
6	=0
7	abort 4805 (terminate immediately if bit 5 = 1, may lose data)/don't abort
8-15	none

NOP command:

Bit #	Meaning if = 1/=0
0	=0
1	=1
2	MODCOMP DI connect/disconnect
3	MODCOMP SI connect/disconnect
4-6	=0
7-15	none

Select command:

Bit #	Meaning if = 1/=0
0-1	=0
2-7	none
8	reset MODCOMP external SI/don't reset
9	reset MODCOMP internal SI/don't reset
10	inhibit 4805 command register load/cause command register load
11	NOVA transmit status/don't transmit status
12	NOVA receive control block/don't receive
13	NOVA receive data block/don't receive
14	unassigned
15	NOVA start autoload/don't start

ODB Ra,MCNV

Output data from Ra to the NOVA. This instruction must be used only when the 4805 is in register mode (set by transfer initiate command), otherwise its effect is undefined. Only one 16-bit word is output each time this instruction is executed.

2.2 NOVA Instructions

DOA ACn,MCNV

Load the address register from ACn. The address register contains the address in NOVA memory where a data channel transfer is to start.

DOB ACn,MCNV

Load the word count register from ACn. The word count register contains the number of words to be transferred in a data channel transfer.

DOC ACn,MCNV

Load the status register from ACn. The status register is an 8-bit register (corresponding to the low-order 8 bits of ACn) which is used to send status information from the NOVA to the MODCOMP. Some of these bits come directly from the 4040 and cannot be set by using a DOC instruction. In addition, only 7 status lines can be read by the 4805, so bit 10 can not be used to send status. However, it can be read and written by the NOVA, and thus could be used for some internal bookkeeping function on the 4040. Other bits are assigned the following meanings (most significant bit = 0):

Bit #	Meaning if = 1/=0
0-7	none
8	4040 power on/off
9	NOVA device error/no error (logical OR of plotter error, software error, and block sequence error)
10	unassigned
11	4040 busy/not busy
12	4040 done/not done
13	NOVA bootstrap request/not requested
14	NOVA load request/not requested
15	transfer out of NOVA/into NOVA

DIA ACn,MCNV

Load ACn from the address register.

DIB ACn,MCNV

Load ACn from the word count register. This can be used to see how a data channel transfer is progressing and whether it is hung or not.

DIC ACn,MCNV

Load ACn from the function and status registers. The function register is an 8-bit register (corresponding to the high-order 8 bits of ACn) which is used to send function commands from the MODCOMP to the NOVA. This instruction also reads the 8-bit status register into the low-order 8 bits of ACn so the NOVA can check the status of its half of the interface. Function register bits are assigned the following meanings (most significant bit = 0):

Bit #	Meaning if = 1/=0
0	4805 power on/off
1	4805 busy/not busy
2	NOVA service interrupt/data interrupt
3	NOVA transmit status/don't transmit
4	NOVA receive control block/don't receive
5	NOVA receive data block/don't receive
6	unassigned
7	NOVA start autoload/don't start
8-15	status bits (see DOC instruction)

S

Start pulse. Sets BUSY = 1, DONE = 0, and begins a data channel transfer as specified in address register, word count register, and in/out status bit.

C

Clear pulse. Sets BUSY = 0, DONE = 0, and stops a data channel transfer if one is in progress. Note that setting DONE = 0 causes an interrupt request in the NOVA. This pulse also clears the status register.

P

P pulse. Cause an external service interrupt request in the MODCOMP.

3. Device Drivers

3.1 MODCOMP-Resident Device Driver

This software is documented in the code itself. Listings are available from Scientific Computing Systems Design Division 2635.

3.2 NOVA-Resident Device Driver

MDEF is the name of a set of NOVA assembly language routines that can be used to drive the MODCOMP-NOVA interface. It should be used in conjunction with the device driver running on the MODCOMP.

External Interface

The user's program interfaces with MDEF through five subroutines and two variables. These routines and variables should all be declared external near the beginning of the user's program with statements such as the following:

```
        .EXTD     .MDEF,.MRMV,.MINT,.DQFUL,.NQFRE
MDEF:   .MDEF
MRMV:   .MRMV
MINT:   .MINT
DQFUL:  .DQFUL
NQFRE:  .NQFRE
        .EXTD     NSER,ERRNO
```

Each of these routines and variables will now be discussed individually.

MDEF

```
Calling sequence:  JSR   @MDEF
                   error return
                   normal return
```

MDEF is always the first call made when using this driver and is used to link the driver to the system (RTOS), initialize buffer data structures, and initiate the task that reports errors. The error return is taken when one of the system calls .TASK, .IDEF, or .STMAP indicates an error (see reference 7).

MRMV

```
Calling sequence:  JSR  @MRMV
                   error return
                   normal return
```

MRMV is always the last call made when using this driver, and is used to remove the driver from the system. The error return is taken when the system call .IRMV indicates an error (see reference 7).

MINT

```
Calling sequence:  JSR   @MINT
                   error return   ; never taken
                   normal return
```

MINT causes an external service interrupt request to be sent to the MODCOMP. This will cause the MODCOMP to check both the hardware and software status of the NOVA. Thus, this is how the NOVA brings error conditions to the attention of the MODCOMP. It is also used to let the MODCOMP know that a buffer has become available (see NQFRE).

DQFUL

```
Calling sequence:  JSR     @DQFUL
                   normal return
```

Returns: AC0 = pointer to buffer node

DQFUL is called whenever the user program wants more data. DQFUL will not return until data is available, and then it returns a single value in AC0. This value is the address of an eight-word long block called a buffer node. The eight words in the node have the following meanings:

Word 1: address of beginning of buffer where data is stored

Word 2: length of the buffer

Word 3: pointer to next node in the list

Word 4: data channel address of beginning of buffer when data is stored

Word 5: reset bit for this buffer

Word 6: block sequence number for this buffer

Word 7: current status of buffer

Word 8: unassigned

NQFRE

```
Calling sequence:  LDA     0,BFNOD  ;BFNOD = pointer to
                                             buffer node.
                   JSR     @NQFRE
                   normal return
```

Returns: AC1 = 1 if buffer available status changed.

NQFRE is called when a buffer that was gotten using DQFUL has been completely processed by the user program and is

ready to be returned. Before calling NQFRE, AC0 must point to the buffer node (exactly as it did on return from DQFUL). When NQFRE returns, the value of AC1 should be checked. If it equals 1, the user program should immediately call MINT to let the MODCOMP know that a buffer has become available.

NSER

NSER should be set to 1 whenever a NOVA software error has occurred. The next time status is transmitted to the MODCOMP, the NOVA software error bit will be set.

ERRNO

ERRNO should be set to some distinct error number (ASCII) whenever a NOVA software error has occurred. The next time status is transmitted to the MODCOMP, this error number will be included as the second word of the transmission.

Internal Structure

The routines in MDEF can be divided into three subgroups according to their functions. These groups and their relations are shown in Figure 3.

The first group is the device handler routines. None of the routines in this group call any MDEF routines outside this group. The six routines in this group are:

MDEF--link device driver to system
MRMV--remove device driver from system
MWRB--write a block from the NOVA to the MODCOMP
MRDB--read a block from the MODCOMP to the NOVA
MINT--send an external service interrupt to the MODCOMP
FSCHK--check the function/status bits on the interface.

The second group is the interrupt handler routines. These routines are entered only upon receiving an interrupt from the MODCOMP-NOVA interface. They are allowed to call any of the other routines in MDEF. The eight routines in this group are:

INTS--interrupt service routine. Control always comes here immediately after receiving a service interrupt from the MODCOMP-NOVA interface. The type of interrupt is decoded, then control is transferred to one of the three following routines:

STST--MODCOMP has requested status. Initiate transmitting status and wait for transfer to complete.

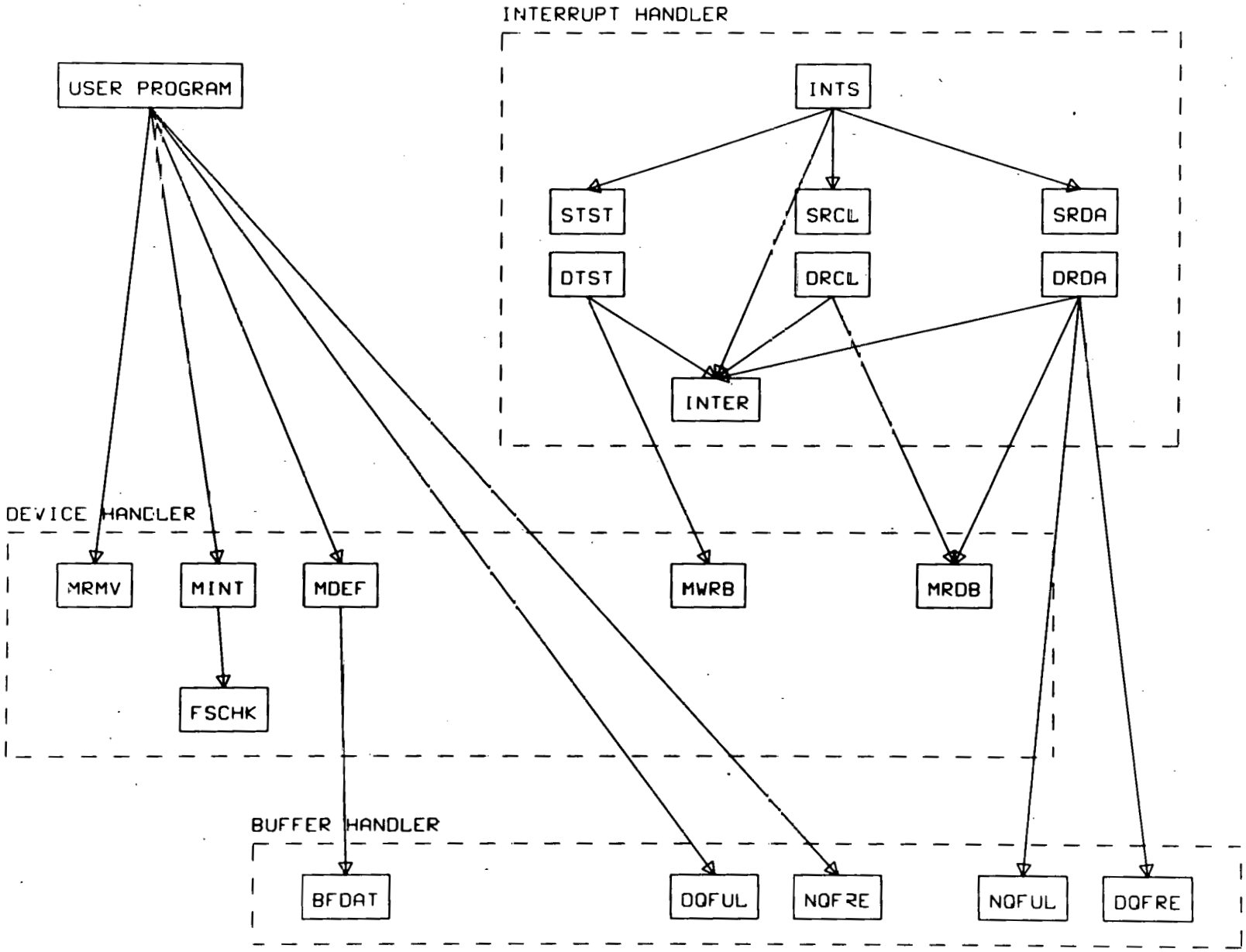

FIGURE 3.
HIERARCHY OF MDEF DEVICE DRIVER ROUTINES

SRCL--MODCOMP wants to send a control block. Initiate receiving control block and wait for transfer to complete.

SRDA--MODCOMP wants to send a data block. Initiate receiving data block and wait for transfer to complete.

Once a transfer has completed, control is transferred to one of the three following routines:

DTST--Clean up after transmitting status.

DRCL--Clean up after receiving control block.

DRDA--Clean up after receiving data block.

If something goes wrong in one of these routines, a special task is initiated which will begin execution once the interrupt has been dismissed:

INTER--handle an interrupt error condition. Output an error message on the plotter and dump NOVA memory.

The status transmitted by STST has the following format:

(Most Significant bit = 0, logic convention: = 1/=0)

Word 1: bit 10: plotter on-line/off-line
bit 11: plotter has paper/out-of-paper
bit 12: plotter ready/not ready
bit 13: NOVA software error/no error
bit 14: block sequence error/no error
bit 15: buffer available/not available

Word 2: error number (ASCII), valid only when bit 13 above = 1

The control block received by DRCL has the following format:

Word 1: word count of following data block

Word 2: reset bit (set on the first block of each graphics file)

Word 3: block sequence number

Words 4-8: unassigned

The third group is the buffer handler routines. These routines manipulate the buffers used to hold data received from the MODCOMP. None of these routines call any others. The buffers are organized as follows:

BUF1, BUF2, BUF3, BUF4--circularly linked buffers of 128 16-bit words in length each

NODE1, NODE2, NODE3, NODE4--buffer nodes of 8 16-bit words in length each.

word 1: address of the corresponding buffer

word 2: length of the buffer

word 3: pointer to the next node in the list

word 4: data channel address of beginning of buffer where data is stored

word 5: reset bit for this buffer

word 6: block sequence number for this buffer

word 7: current status of buffer

word 8: unassigned

FREE--contains a pointer to the node of the next free buffer.

FULL--contains a pointer to the node of the next full buffer.

The five buffer handler routines are:

DQFUL--get the buffer node pointed to by FULL. If its buffer status is not "full," then wait till it is. Then change status to "processing," move FULL pointer to the next node and return a pointer to the "processing" node.

NQFRE--change status of the buffer node passed in to "free" and indicate buffer available. If no buffers were previously available, return AC1 = 1.

DQFRE--get the buffer node pointed to by FREE. If its buffer status is not "free," then return an error. Else change status to "reading," move FREE pointer to the next node, and return a pointer to the "reading" node. Also, if this was the last "free" buffer, indicate no buffers available.

NQFUL--change status of the buffer node passed in to "full."

BFDAT--initialize the buffer data structures (see Figure 4).

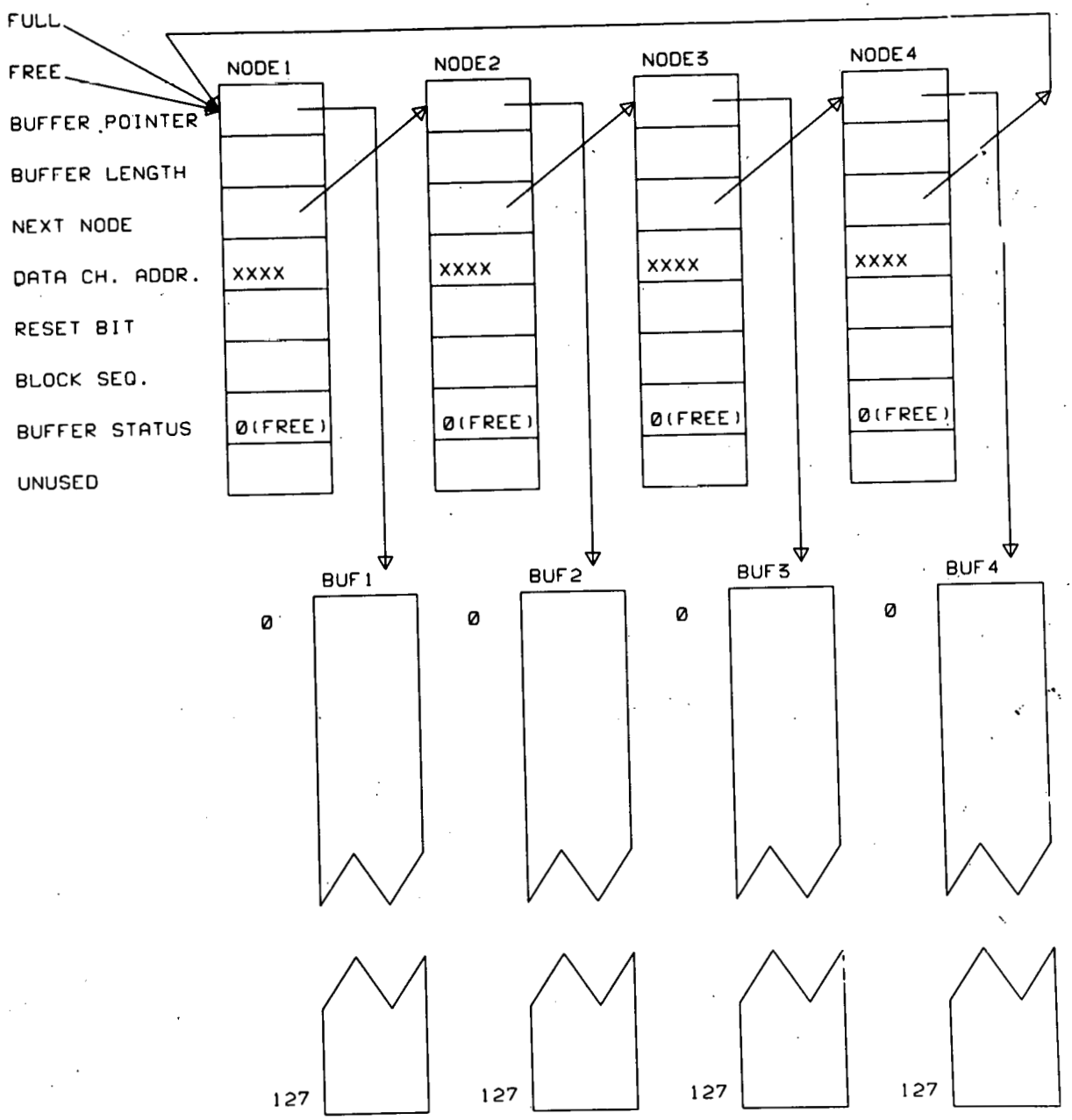

FIGURE 4.

BUFFER DATA STRUCTURES AFTER INITIALIZATION

3.3 MODCOMP-NOVA Protocol

Using the MODCOMP-NOVA interface with the device drivers described earlier will cause communications to be carried on between the MODCOMP and NOVA as follows:

Transmit Data Block:

MODCOMP	NOVA
Test hardware status. If not OK, give diagnostic and keep trying.	
Send hardware command "Transmit Status"	Interrupt
Start DMA input of software status	Start DMA output of software status
.	.
.	.
.	.
Transfer complete (If time out, give diagnostic and quit.)	Transfer complete
If software status not OK, take appropriate action.	
Send hardware command "Receive Control Block"	Interrupt
Start DMA output of control block	Start DMA input of control block
.	.
.	.
.	.
Transfer complete (If time out, give diagnostic and quit).	Transfer complete
Send hardware command "Receive Data Block"	Interrupt
Start DMA output of data block	Start DMA input of data block
.	.
.	.
.	.
Transfer complete (If time out, give diagnostic and quit.)	Transfer complete

Autoload:

MODCOMP	NOVA
Test hardware status. If not OK, give diagnostic and keep trying.	
Send hardware command "Start Autoload"	Cause actions equivalent to hitting console switches: STOP,RESET,& PR LOAD. PR LOAD causes execution of code in autoload ROM.
Interrupt	Send hardware status "Bootstrap request"
Start DMA output of bootstrap (400_8 words) . . .	Start DMA input of bootstrap. . . .
Transfer complete (If time out, give diagnostic and quit.)	Transfer complete.
	Execute bootstrap.
Interrupt	Send hardware status "Load request"
Start DMA output of system and programs . . .	Start DMA input of system and programs . . .
Transfer complete (If time out, give diagnostic and quit.)	Transfer complete
Wait. Check "Load request"	Execute system and programs. Clear "Load request"

4. Diagnostics

4.1 MODCOMP-Resident Diagnostics

This software is documented in the code itself. Listings are available from Scientific Computing Systems Design Division 2635.

4.2 NOVA-Resident Diagnostics

This program is intended to provide extensive functional testing of the MODCOMP-NOVA interface from the NOVA side. Tests are selected by setting corresponding console switches. Running the first five tests of this program requires a NOVA minicomputer with 8K words of memory, the NOVA half (4040 board) of a MODCOMP-NOVA interface, and a line printer (e.g., electrostatic printer/plotter). It is assumed that the MODCOMP half of the interface (4805 board) will remain inactive during these tests. Running the last two tests requires a MODCOMP minicomputer and corresponding diagnostic software in addition to the above. This program runs under RTOS.

The console switches are used to alter the execution of the tests and to select the tests to be executed. Settings are as follows (MSB = 0):

```
 0 - halt on error
 1 - inhibit error printout
 2 - inhibit success printout
 3 - loop in test
 6 - read status (test 1)
 7 - write and read status (test 2)
 8 - write and read address register (test 3)
 9 - write and read word count register (test 4)
10 - hung transfer (test 5)
11 - receive and echo block (test 6)
12 - interrupt and receive command (test 7)
```

The diagnostic program is started by loading it (MDIAG) off the disk and running it. The first thing it does is halt and wait for you to set the desired console switches. Then press CONTINUE, and the program will begin execution. The program can always be restarted in this manner whenever it halts.

For each test that is completed successfully, the message "TEST n SUCCEEDED" is output, where n is the test number. For each test that fails, the message "TEST n FAILED" is output, where n is the test number. Also, an error message of the form "i SHOULD BE j" is output, where i is a value read from some register on the interface, and j is what that value should be.

The individual tests are described in more detail below:

Test 1: Read Status--Check status of the interface while it is idle.

Test 2: Write and Read Status--Checks that all writeable bits in the function/status register can be written and read correctly.

Test 3: Write and Read Address Register--Checks that various bit patterns can be written to and read from the address register correctly.

Test 4: Write and Read Word Count Register--Checks that various bit patterns can be written to and read from the word count register correctly.

Test 5: Hung Transfer--Initiates an input transfer which should hang, since the MODCOMP shouldn't be sending anything. Status is checked both before and after terminating the transfer.

Test 6: Receive and Echo Block--Initiates an input transfer of 256 words. The corresponding MODCOMP diagnostic should send a block of 256 words. Upon completion of the transfer, the contents of the block are printed on the NOVA plotter (if console switch 1 is not set) and status is checked. Then the diagnostic halts (if console switch 3 is not set) and will continue when the console switch CONT is pressed. An output transfer of the same 256 words just received is initiated. The corresponding MODCOMP diagnostic should receive a block of 256 words and check that block against the one just sent. Upon completion of the transfer, status is checked.

Test 7: Interrupt and Receive Command--An interrupt is sent to the MODCOMP, where corresponding diagnostic software should be expecting it. The MODCOMP should then send a command with none of the function bits set. This will cause an interrupt in the NOVA which will then check status. This sequence is repeated 16 times to check every possible setting of the four function bits.

References

1. Data Terminal/Computer Link, Model 4805-1/Model 4820, Technical Manual, Pub. No. 225-200125-001, MODCOMP, Aug. 1977.

2. MODCOMP II Computer Reference Manual, Pub. No. 210-102000-000, MODCOMP, Oct. 1973.

3. Data General User's Manual, Interface Designers Reference, NOVA and Eclipse Line Computers, Ordering No. 015-000031, Data General Corp., Feb. 1978.

4. Data General Technical Manual, NOVA 3, ordering No. 015-000053, Data General Corp., Dec. 1976.

5. Data General User's Manual, Programmer's Reference, NOVA Line Computers, Ordering No. 015-000023, Data General Corp., July 1977.

6. Data General User Device Driver Implementation in the Real-Time Operating System, Application Note, Order No. 017-000006, Data General Corp., April 1977.

7. Data General Reference Manual, Real Time Operating System, Ordering No. 093-000056, Data General Corp., Feb. 1975.

RWS:pml:1708A:P

Distribution:
2635 P. A. Lemke
2635 R. E. Domres
2635 J. Brabson
2640 J. L. Tischhauser
2644 D. M. Darsey
2644 R. W. Simons (10)
2644 C. D. Brown
2648 D. H. Schroeder
2648 L. G. Pierson
8266 E. E. Aas
3141 T. L. Werner (5)
3151 W. L. Garner (3)
for DOE/TIC
DOE/TIC (25)
(R. P. Campbell, 3154-3)

Org.	Bldg.	Name	Rec'd by*	Org.	Bldg.	Name	Rec'd by*

*Recipient must initial on classified documents.